G-Tech : Jurnal Teknologi Terapan
Vol. 10, No. 2, April 2026, p. 652-665
E-ISSN: 2623-064X | P-ISSN: 2580-8737

Privacy-Focused AIoT: Implementing an Offline Voice Assistant for Smart Building Management Using Local LLMs

Fitri Wibowo[1]✉, Suheri[2], Ferry Faisal[3], Freska Rolansa[4]

[1, 2, 3, 4] Informatics Engineering, Electrical Engineering Department, Politeknik Negeri Pontianak, Indonesia

✉ **Corresponding Author :** Fitri Wibowo (e-mail: fitri.wibowo@polnep.ac.id)

Article Information

Article History

Received : February 20, 2026
Revised : March 02, 2026
Published : April 04, 2026

***Keywords*:**

AIoT;
Edge computing;
Local large language model;
Offline voice assistant;
Smart building

ABSTRACT

Voice assistants are increasingly used for smart building control, yet cloud-based architectures raise privacy risks and become unavailable during internet outages. This study designs and evaluates a fully offline AIoT voice assistant for smart building management using local speech and language models. The system employs an edge audio node (Raspberry Pi Zero 2W with ReSpeaker 2-Mics Pi HAT) and a local GPU server running containerized microservices for speech-to-text (Whisper), intent understanding and action planning (Ollama-hosted LLMs), and text-to-speech (Piper). Building devices and sensors are integrated through Home Assistant, enabling voice-driven control and monitoring without sending audio or interaction logs to external services. Experiments in a laboratory smart-building testbed evaluate speech recognition robustness under varying noise levels, LLM command understanding accuracy and memory footprint, and end-to-end IoT task execution. The speech subsystem achieves a Word Error Rate of 5–20% depending on background noise. Across 33 IoT entities, the assistant reaches a 96.67% execution success rate with an average response time of 5.5 s. Among the evaluated local models, Qwen3 8B achieves the highest intent-to-action accuracy (Acc_I2A=100% on an oracle-text command test set with N=43) with 6.8 GB memory use. The results demonstrate that privacy-preserving and resilient voice interaction for smart building management is feasible using current local LLM stacks.

INTRODUCTION

The convergence of Internet of Things (IoT), Artificial Intelligence (AI), and edge computing technologies has fundamentally transformed the landscape of smart building management. The concept of Artificial Internet of Things (AIoT)—which synergizes AI with IoT infrastructure—enables connected devices to process data autonomously and make intelligent decisions at the network edge without relying on centralized cloud services (Zhou et al., 2019). Within this ecosystem, voice assistants have emerged as crucial natural language interfaces that facilitate intuitive human-machine interaction, particularly for control and management tasks in complex building environments. The latest advances in Large Language Models (LLMs)—sophisticated neural networks capable of understanding and generating human-like text—have opened unprecedented opportunities to enhance the contextual awareness and decision-making capabilities of voice-based building management systems (Dwarampudi & Yogi, 2024; Shen et al., 2023). Understanding user perspectives on data privacy remains fundamental to developing effective voice assistant systems. Research has demonstrated significant variability in user privacy concerns (Maier et al., 2023), with some users willing to adopt voice assistant technologies when

they perceive tangible benefits, yet the majority maintain strong preferences for personal data protection (Maier et al., 2023). This diversity in privacy awareness necessitates careful consideration of architectural choices that balance functionality with user data sovereignty.

Contemporary smart building solutions predominantly depend on cloud-based voice assistant architectures (such as Amazon Alexa, Google Assistant, or Apple Siri), which present formidable privacy and security vulnerabilities. These centralized systems routinely transmit sensitive data—including user voice patterns, behavioral preferences, activity logs, and security-related information—to remote data centers operated by third-party providers. Security and privacy challenges of virtual assistants remain inadequately addressed, particularly regarding unauthorized data collection practices and insufficient security measures in cloud-based ecosystems (Bolton et al., 2021; J. Li et al., 2023; Yan et al., 2023). Such practices expose users to multiple risks, including data breaches, unauthorized access, and potential misuse of personal information. Attack surface analysis of commercial voice assistants reveals persistent vulnerabilities in authentication, access control, and data exposure (Y. Li et al., 2021).

Furthermore, cloud-dependent systems exhibit critical operational fragility; they become non-functional during inter-net outages or network disruptions (Gondi & Pratap, 2021), rendering building management capabilities unavailable precisely when buildings face emergency situations requiring automated responses. A secondary limitation of existing voice assistant implementations is their constrained natural language understanding. Most deployed systems rely on rigid rule-based or shallow machine learning approaches rather than modern LLMs, resulting in limited contextual comprehension and inflexibility when interpreting complex or non-standard commands.

Integrating AI with IoT technology introduces multiple challenges in security, device interoperability, and privacy preservation. Research has identified critical needs for privacy-preserving solutions that employ edge computing and secure data analysis methods to deploy local language models on voice assistant systems, thereby improving both privacy and overall system performance (Dwarampudi & Yogi, 2024; Shen et al., 2023). Additionally, longitudinal studies of user interaction patterns reveal evolving perspectives on privacy and the multifaceted nature of how users engage with AI voice assistants—ranging from viewing them as purely functional tools to developing relational expectations (Xu & Li, 2024). These findings underscore the necessity for research-driven approaches to privacy-aware voice assistant design that account for dynamic user expectations.

The aforementioned limitations motivate this research, which addresses two primary research questions: (1) How can we develop a fully offline AIoT voice assistant system capable of operating independently from cloud infrastructure while maintaining sophisticated natural language understanding through local LLMs? (2) What is the achievable performance of such a system in managing smart building operations through voice commands, particularly regarding speech recognition accuracy, inference latency, resource consumption across the edge audio endpoint and the on-premises inference server, and integration with heterogeneous IoT devices?

This research develops and implements an offline AIoT voice assistant system that leverages locally-deployed Large Language Models for comprehensive smart building management. The system architecture prioritizes three critical design objectives: (1) privacy-by-design through exclusive local data processing, ensuring user data remains within the organizational boundary; (2) operational resilience through complete independence from internet connectivity, enabling uninterrupted service during network disruptions; and (3) intelligent command interpretation through LLM-based natural language understanding, supporting nuanced and context-aware building control. The proposed solution utilizes Raspberry Pi Zero 2W with a ReSpeaker 2-Mics Pi HAT as the edge computing node for audio acquisition (Arora et al., 2025), while integrating GPU-accelerated local servers running containerized microservices (Whisper for speech-to-text, Ollama for LLM inference, and Piper for text-to-speech) alongside Home Assistant as the IoT integration platform (Birkmose et al., 2025).

This study makes four main contributions. First, we present an end-to-end offline AIoT voice assistant for smart building management using a two-tier deployment consisting of an edge audio endpoint and an on-premises inference server. The architecture is designed to operate without external cloud services by keeping speech-to-text (STT), LLM inference, and text-to-speech (TTS) fully inside a local intranet. Second, we implement a containerized service pipeline that integrates Whisper for STT, an Ollama-hosted local LLM for intent understanding and action planning, and Piper for TTS. This pipeline is orchestrated through Home Assistant's built-in local REST API and voice pipeline to translate speech transcripts into device/service actions. Third, we evaluate speech recognition robustness in a laboratory smart-building testbed using Word Error Rate (WER) under multiple background-noise conditions. Finally, we benchmark local LLMs within the same command-understanding pipeline and report command understanding accuracy on the study test set, model memory footprint, and end-to-end performance in terms of IoT task execution success rate and response latency across 33 Home Assistant entities.

This research addresses a critical need in smart building technology as escalating privacy incidents, documented vulnerabilities in centralized cloud systems, and rising user awareness increase demand for privacy-preserving and resilient alternatives (Maier et al., 2023), especially as smart building deployments expand across educational institutions, corporate offices, and public facilities. The novelty of this work lies in (i) designing and implementing a comprehensive, fully offline AIoT architecture optimized for smart building management (departing from cloud-dependent approaches), (ii) systematically configuring and benchmarking state-of-the-art local LLMs (Qwen3, Llama3.1, and Mistral 7B) for offline command understanding and intent-to-action mapping on an on-premises GPU server with accuracy and memory-footprint characterization, (iii) empirically evaluating performance across multiple acoustic environments and heterogeneous IoT device categories in an integrated smart building ecosystem, and (iv) developing a containerized microservices architecture that supports reproducibility, maintainability, and scalability across deployment contexts; this is practically validated in the Informatics Engineering Laboratory of Politeknik Negeri Pontianak with immediate utility for laboratory automation and environmental monitoring.

The evaluated LLMs were chosen because they are readily available in Ollama and support reliable structured outputs/tool-calling behavior for producing Home Assistant service-call arguments, while remaining feasible to run on the available on-premises GPU at the 7B--8B parameter scale.

Prior studies highlight the role of edge computing for AIoT deployments that require low latency and data sovereignty, and note that resource-efficient techniques are increasingly enabling transformer-based models on constrained devices (Z. Li et al., 2024; Ye et al., 2024; Yu et al., 2024; Zheng et al., 2025). In parallel, research on user adoption emphasizes that privacy perceptions vary widely and may evolve over time, which motivates privacy-aware assistant designs for institutional smart-building contexts (Maier et al., 2023; Xu & Li, 2024).

Security analyses of commercial voice assistant ecosystems continue to report persistent risks related to data exposure and ecosystem integrations, strengthening the case for local-only processing and administratively controllable deployments. While federated learning can support privacy-preserving model improvement, its practical trade-offs motivate an engineering focus on robust offline operation and measurable end-to-end performance in this study (Pelikan et al., 2025).

Overall, the literature motivates an end-to-end, fully offline stack that integrates local speech processing, local LLM inference, and heterogeneous device control under a unified local automation platform, and it informs the evaluation dimensions used in this work.

RESEARCH METHODS

Research Design and Setting

This work follows a design-and-implementation approach with experimental evaluation in a real smart building testbed deployed at the Informatics Engineering Laboratory, Politeknik

Negeri Pontianak. The focus is to implement a voice-driven AIoT assistant that operates fully offline (no external cloud dependency) and to evaluate its performance in speech recognition, local LLM-based command understanding, end-to-end latency, resource usage, and IoT device integration reliability.

System Architecture

The assistant is implemented as a fully offline, two-tier architecture consisting of an edge audio endpoint and a local server that hosts containerized AI services and the building automation platform. All communication occurs within the local intranet; no user audio, transcripts, or interaction logs are transmitted to external services. Figure 1 summarizes the functional pipeline and end-to-end message flow.

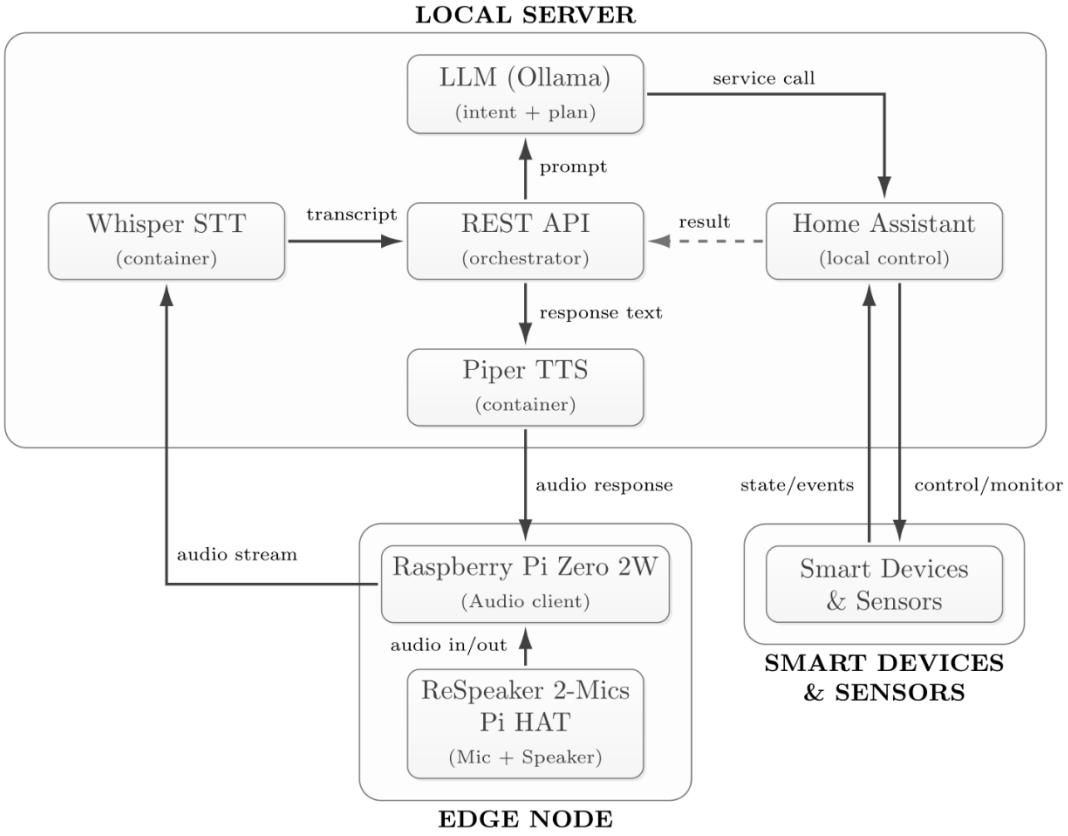

Figure 1. Functional architecture and end-to-end processing pipeline.

As shown in Figure 1, the proposed implementation can be summarized into three functional layers. Each layer is characterized below in terms of its role and the primary hardware/software components used in this study.

1. Edge node (audio endpoint).
   Raspberry Pi Zero 2W + ReSpeaker 2-Mics Pi HAT for audio capture and play-back.
2. Local inference server (AI microservices).
   Whisper STT, Ollama LLM inference, and Piper TTS, deployed as containers on a GPU-capable Ubuntu server.
3. Orchestration and IoT integration layer.
   Home Assistant exposes smart-building entities and orchestrates the pipeline through its built-in local REST API.

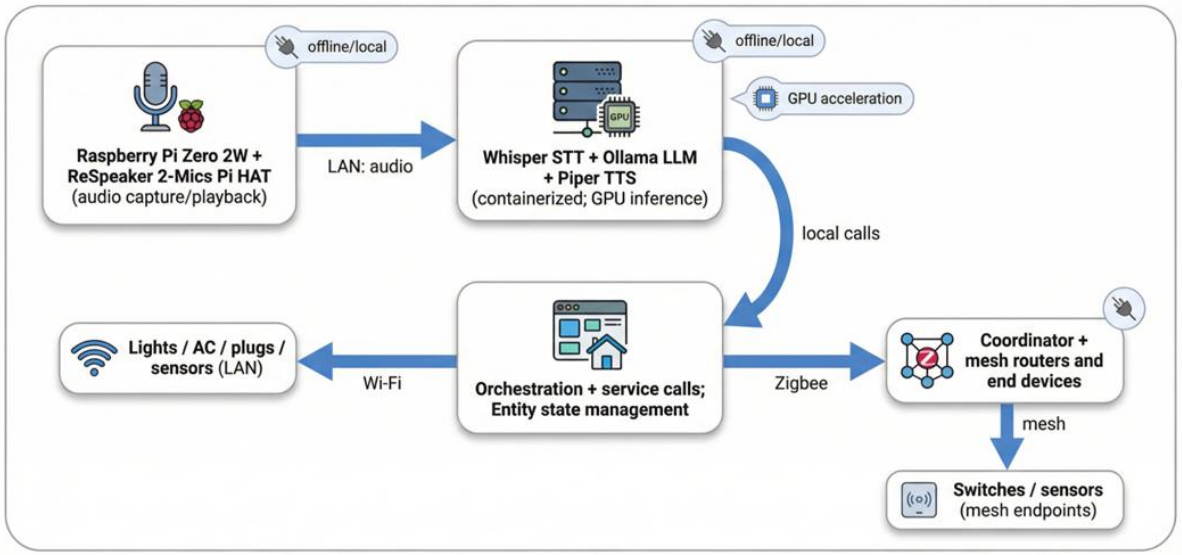

Figure 2. Physical testbed topology (experimental setup) of the offline voice assistant within the local network (LAN).

Figure 2 depicts the evaluated experimental setup: an edge audio endpoint streams audio over the local network (LAN) to an on-premises GPU server hosting the STT/LLM/TTS microservices, while Home Assistant performs local orchestration and integrates heterogeneous smart-building devices via Wi-Fi and Zigbee. In contrast, Figure 1 abstracts the same system at a functional level to highlight the processing stages and message flow.

Hardware and Software Configuration

To support reproducibility, Table 1 summarizes the main hardware and software components used in the prototype and evaluation.

The table details the platform specifications (edge endpoint and GPU server), fixed software versions, and the key inference settings used for the STT/LLM/TTS services and Home Assistant integration. Unless stated otherwise, these settings were kept fixed across experiments to ensure that reported performance differences reflect model and workload effects rather than untracked environment changes.

Table 1. System configuration for reproducibility

Component	Specification
Edge audio endpoint	Raspberry Pi Zero 2W; ReSpeaker 2-Mics Pi HAT; OS: Raspberry Pi OS (Legacy, 64-bit; installed via Raspberry Pi Imager)
Local inference server	Ubuntu 24.04.2 LTS; CPU: Intel Xeon E5-2697 v4 @ 2.30 GHz; RAM: 32 GB (total); GPU: NVIDIA GeForce RTX 3060 (12 GB VRAM); storage: 120 GB SSD (mounted at /mnt/data-ssd) + 250 GB SSD (system)
STT service	Wyoming Whisper container image: fitriwibowo/wyoming-whisper:2.2.0; model: medium-int8; language: id; beam size: 5; device: CUDA
LLM service	Ollama 0.16.1 (Docker); Qwen3 8B (qwen3:8b, Q4_K_M, context length 40960); Llama 3.1 8B (llama3.1:8b, Q4_K_M, context length 131072); Mistral 7B (mistral:latest, Q4_0, context length 32768)
TTS service	Wyoming Piper container image: rhasspy/wyoming-piper; voice: en_US-lessac-medium; sample rate: 22050 Hz; Piper version: 1.0.0
Smart building platform	Home Assistant Core 2025.12.5; integration protocols: Wi-Fi and Zigbee
Containerization	Docker Engine - Community 28.2.2; Docker Compose version: v2.36.2

Note: The listed software versions and inference settings reflect the fixed configuration used during the experiments in this study.

We selected LLM candidates that are supported by Ollama and can be constrained to produce structured outputs (e.g., JSON) or tool calls (via the chat API tools mechanism) (Ollama, 2026) so that voice transcripts can be mapped into Home Assistant service calls in a deterministic and auditable way. We focused on 7B--8B class models because they fit the available RTX 3060 (12 GB VRAM) under 4-bit quantization and provide a practical balance between latency, memory footprint, and command-understanding accuracy for an on-premises deployment.

Voice Command Processing Pipeline

The interaction pipeline consists of: (1) audio acquisition on the edge node; (2) speech recognition via Whisper (Andreyev, 2025; Zhang et al., 2025) to generate text commands; (3) prompt construction and intent interpretation using a local LLM served by Ollama; (4) mapping the interpreted intent to Home Assistant service calls (e.g., device control, state queries); and (5) generating spoken feedback using Piper TTS (Vecino et al., 2023) and returning audio output to the edge node. Containerization is used to isolate services, simplify deployment, and enable reproducible configuration (Beňo et al., 2024; Q. Li et al., 2025). In the current prototype, spoken feedback uses an English (*en_US*) Piper voice; developing an Indonesian TTS pipeline is left for future work.

In this manuscript, REST API (orchestrator) denotes the Home Assistant-provided local REST interface and voice pipeline that coordinate service calls and responses; no additional standalone orchestrator component was developed as part of this work.

Figure 3 illustrates the runtime interaction sequence among the edge audio endpoint, on-premises AI services, Home Assistant orchestration, and smart-building devices. The diagram also indicates the stage-level timestamp boundaries (t_0–t_5) used for latency instrumentation (Section Latency Instrumentation).

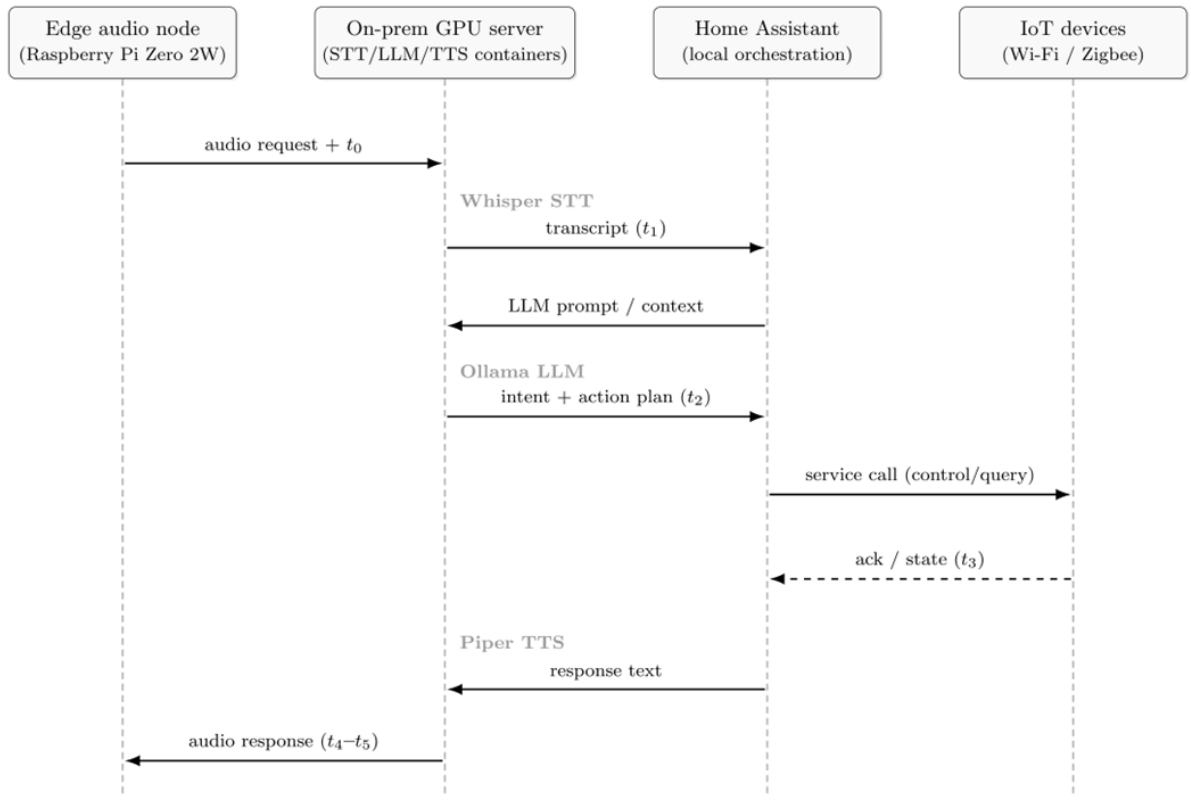

Figure 3. Sequence diagram of the offline voice assistant runtime interaction within the local network (LAN), highlighting the timestamp boundaries t_0–t_5 used for stage-level latency instrumentation.

Figure 3 highlights the key step that bridges the LLM output to building control: Home Assistant converts the returned *action plan* into an executable service call. In our pipeline, the LLM is constrained to produce a structured result (e.g., a JSON object or a tool-call payload) that specifies the target Home Assistant service (domain and service name) and the required call data (primarily **`entity_id`**, and optionally **`service_data`**). For example, an instruction to turn on the Lecturer Room 2 AC is mapped to the service **`switch/turn_on`** with **`entity_id = switch.ac_r_dosen_2`**; similarly, unlocking the main door is mapped to **`lock/unlock`** with **`entity_id = lock.front_door_it_lab`**. Home Assistant treats this payload as an input to its local orchestration layer: it validates the fields against an allowlisted set of supported services/entities and then dispatches the request via the internal service registry (equivalently, via the local REST endpoint **`/api/services/<domain>/<service>`** with a JSON body containing **`entity_id`** and **`service_data`**). For monitoring queries, Home Assistant performs a local state lookup (e.g., reading **`sensor.termometer_r_server_temperature`**) and returns the current state to the pipeline. After execution/query completes, Home Assistant returns a concise outcome (success/failure and optional state) so that the TTS subsystem can produce the spoken feedback.

Operational Definition of Fully Offline

In this paper, fully offline is defined operationally as follows: during steady-state operation, the assistant performs (i) speech recognition, (ii) command interpretation/planning, (iii) smart-building action execution, and (iv) speech synthesis using only on-premises compute and local-network communication, without requiring external cloud APIs for functionality.

To avoid ambiguity, Table 2 summarizes an offline-compliance checklist describing the criteria implied by this definition and how they relate to the current prototype and evaluation scope.

Table 2. Offline-compliance checklist (operational definition and evaluation scope)

Criterion	Rationale	Status in this study
No cloud dependency for STT/LLM/TTS	Prevents audio/transcript leaving the premises and avoids internet outage failures	Implemented by local Whisper, Ollama, and Piper services on the on-premises server
Local-only orchestration and device control	Ensures building actions remain functional within the LAN	Implemented via Home Assistant local REST API and service execution
Models and containers provisioned ahead of runtime	Prevents implicit downloads/updates at inference time	Required for offline operation; not formally audited in this manuscript
No outbound network traffic required at runtime (DNS/telemetry/updates)	Avoids unintentional data leakage channels	Not formally verified with packet-capture; treated as a limitation and deployment requirement
Local logging and retention policy defined	Clarifies whether transcripts are stored and for how long	Not fully specified; future work will formalize retention and access control

Security Considerations

Voice-driven building control introduces safety and security relevant risks because natural language inputs can trigger privileged actions (e.g., unlocking doors or enabling power circuits). In the context of an offline deployment, the primary threat surface shifts from cloud exposure to local misuse and intranet compromise. The following considerations guide the prototype design and are recommended for operational deployment.

1. Action allowlisting and schema validation.
   The assistant integration (implemented via Home Assistant) should restrict executable actions to an allowlisted set of Home Assistant services and entities, and validate LLM

outputs against a strict schema before issuing any service calls. This reduces the impact of hallucinated or malformed plans.

2. Least-privilege credentials.
   Home Assistant access tokens/API keys should be scoped to the minimum privileges required for the evaluated entities, stored outside source code (e.g., environment variables or secrets management), and rotated periodically.
3. Authorization for sensitive intents.
   High-impact commands (e.g., door unlock) should require additional authorization (e.g., local user authentication, voice PIN, or second-factor confirmation) and should be excluded by default in shared environments.
4. Prompt-injection and transcription attacks.
   Because spoken content can include adversarial instructions, the prompting and assistant policy should treat transcripts as untrusted input (e.g., refuse requests to reveal secrets, modify system prompts, or execute non-allowlisted actions).
5. Network security and auditability.
   Communication between services on the intranet should use authenticated channels where feasible (e.g., TLS), with local audit logs for issued actions and outcomes to support incident investigation.

These aspects are discussed as design requirements; a full security evaluation (penetration testing, formal threat modelling, and adversarial robustness) is outside the scope of the current performance-focused experiments.

Data Collection and Evaluation Metrics

System performance is assessed using multi-metric measurements collected via service logs and local monitoring tools (e.g., Docker logs, Home Assistant debugging tools, and GPU monitoring). The main metrics include:

1. Speech recognition.
   Word Error Rate (WER) measured under multiple acoustic conditions (quiet, low noise, high noise) to reflect realistic environments.
2. LLM performance.
   Command understanding accuracy (intent-to-action correctness), inference time/latency, and memory usage during inference.
3. End-to-end system performance.
   Overall response time from speech input to action execution and spoken response; CPU and RAM utilization during continuous operation.
4. IoT integration reliability.
   Success rate and response time for representative operations such as device control and sensor reading across heterogeneous entities.

Audio Acquisition Setup and WER Computation

Speech recognition robustness is evaluated using Word Error Rate (WER) under three background-noise conditions (quiet, low noise, high noise). Audio is captured using the ReSpeaker 2-Mics Pi HAT attached to the Raspberry Pi Zero 2W. During recording, the speaker stands at an approximate distance of 1.0 m from the microphone in the same room configuration across conditions. All recordings are conducted in an enclosed laboratory room with typical indoor reverberation; no dedicated acoustic treatment is applied. To reduce confounding factors, speaker orientation, microphone placement, and room layout are kept constant across conditions.

Recordings are stored on the edge node in an uncompressed audio format (e.g., PCM/WAV) using a fixed sampling configuration that is held constant for all WER measurements. Ambient noise level is monitored using a smartphone SPL meter application. The reported dB ranges (Table 3) should therefore be interpreted as approximate indicative ranges rather than laboratory-calibrated sound pressure level measurements.

For each acoustic condition, we record 10 utterances from a single speaker (total of 30 utterances). Ground truth transcripts are produced manually by listening to the recordings and transcribing the intended content.

WER is an ASR-only metric and does not directly indicate end-to-end device-control correctness; in the full pipeline, some word-level transcription errors may still preserve the intended intent/slots and thus lead to successful execution.

WER is computed using the standard definition based on edit distance between the reference transcript and the ASR hypothesis:

$$\mathrm{WER} = \frac{S + D + I}{N_{\mathrm{ref}}} \times 100\% \quad (1)$$

Throughout this paper, WER is computed according to equation (1). where S, D, and I denote the number of substitutions, deletions, and insertions, respectively, and N_{ref} is the number of words in the reference transcript.

Command Understanding Test Set and Ground Truth

To evaluate LLM-based command understanding using a clearly specified and reproducible protocol, we use a predefined test set comprising $N = 43$ voice commands (utterances) that cover both control (e.g., turning switches/breakers on/off, locking/unlocking) and query intents (e.g., reading temperature/weather and door status) across the Home Assistant entities used in this study. Each test utterance is annotated with a ground-truth target action represented as a tuple $a = (s, e, p)$, where s denotes the Home Assistant service (domain and service name), e denotes the target entity identifier, and p denotes any required parameters.

To make the dataset more concrete and to reflect our design objective of nuanced, context-aware building interaction, the test utterances explicitly reference locations (rooms/floors) and common aliases used by occupants. Examples of the speech used in the command-understanding test set (verbatim) include: *"What is the status of the Lecturer Room 2 AC?"*, *"Turn on/off the Lecturer Room 2 AC"*, *"Is the main door locked or unlocked?"*, *"What is the temperature in Ruang Server?"*, and *"What's the weather like?"*. These examples require the assistant to resolve context (e.g., room names such as *Ruang Server*) and aliases (e.g., *main door*) into the correct Home Assistant entity and service call, rather than relying on fixed keyword matches.

Command Understanding and Success Metrics

We distinguish between correctness of the interpreted action and success of the subsequent execution. To isolate LLM behaviour from ASR errors while still reporting end-to-end realism, command understanding is evaluated under two input conditions: (i) oracle text (manually verified transcripts) and (ii) STT transcripts (Whisper output). Unless stated otherwise, model-comparison accuracy refers to the oracle-text condition.

1. Intent-to-action accuracy ($\mathrm{Acc}_{\mathrm{I2A}}$)
   The fraction of test utterances for which the pipeline results in a Home Assistant service call that matches the ground-truth action tuple (service, entity ID, and parameters). Formally,

$$\mathrm{Acc}_{\mathrm{I2A}} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}\left[\hat{a}_i = a_i\right]. \quad (2)$$

2. Execution success rate ($\mathrm{Succ}_{\mathrm{exec}}$)
   The fraction of attempts for which the issued service call is accepted by Home Assistant and the target entity reports successful state transition or a valid query response within a timeout.

Latency Measurement and Breakdown

End-to-end response time is measured at the system level as the elapsed time from the end of speech on the edge node (i.e., VAD end or stop-recording event) until the edge node receives the corresponding synthesized audio response after the Home Assistant action/query has been processed. This end-to-end metric captures both inference and orchestration overhead in a practical deployment. To facilitate bottleneck analysis, the end-to-end time can be decomposed into the following stages:

- T_{STT}: speech-to-text processing time (Whisper).
- T_{LLM}: LLM inference time for intent interpretation and action planning (Ollama).
- T_{HA}: Home Assistant service-call time (including entity state retrieval for queries).
- T_{TTS}: text-to-speech synthesis time (Piper).
- T_{net}: intra-LAN request/response overhead between components.

Accordingly, $T_{\mathrm{e2e}} \approx T_{\mathrm{STT}} + T_{\mathrm{LLM}} + T_{\mathrm{HA}} + T_{\mathrm{TTS}} + T_{\mathrm{net}}$.

Latency Instrumentation

To obtain per-stage latency values, the Home Assistant pipeline and the invoked services record timestamps at key boundaries and compute stage durations using differences between successive timestamps. In our implementation, the edge node provides the end-of-speech timestamp (t_0) together with the audio request; the server-side services then log: t_1 (STT complete), t_2 (LLM complete), t_3 (Home Assistant service call complete), t_4 (TTS complete), and t_5 (audio response returned to the edge). Stage-level latencies are computed as $T_{\mathrm{STT}} = t_1 - t_0$, $T_{\mathrm{LLM}} = t_2 - t_1$, $T_{\mathrm{HA}} = t_3 - t_2$, $T_{\mathrm{TTS}} = t_4 - t_3$, $T_{\mathrm{net}} = t_5 - t_4$, and $T_{\mathrm{e2e}} = t_5 - t_0$.

Evaluation Protocol

Experiments are organized into four groups: (1) speech recognition robustness across acoustic conditions; (2) comparison of multiple local LLM candidates (Qwen3 8B, Llama3.1 8B, and Mistral 7B) (Aydin et al., 2025; Jiang et al., 2023) for command understanding efficiency and resource usage; (3) system-level tests such as cold-start and continuous operation to observe stability and resource consumption; and (4) integration tests through Home Assistant to validate compatibility with diverse smart building devices and sensors. Based on this protocol, the empirical results are reported in the next section.

RESULTS AND DISCUSSION

This section presents the empirical results of the offline AIoT voice assistant prototype, covering (i) speech recognition robustness, (ii) local LLM model selection, and (iii) smart building integration reliability.

We report Word Error Rate (WER) computed using equation (1), intent-to-action accuracy using equation (2), and end-to-end integration performance in terms of execution success rate and response time.

Speech Recognition Performance

Speech recognition was evaluated under three acoustic conditions to reflect realistic operational environments. As expected, recognition accuracy degraded as background noise increased, in line with prior findings that environmental noise and reduced SNR substantially increase word error rates in automatic speech recognition systems (Khan et al., 2025). A key mechanism is reduced signal-to-noise ratio (SNR): background noise masks low-energy phonetic cues and degrades spectral features used by the ASR model, which increases substitution and deletion errors in the decoded transcript. In addition, higher noise can blur the effective speech boundaries for voice activity detection/segmentation and introduce short non-speech fragments, leading to more insertions and overall higher WER.

Table 3. Speech recognition results across acoustic conditions

Acoustic condition	WER (%)
Quiet room (0–20 dB)	5
Low noise (30–40 dB)	10
High noise (50–60 dB)	20

Note: The dB ranges are approximate indicative values monitored using a smartphone SPL meter and should not be interpreted as laboratory-calibrated SPL measurements.

Local LLM Model Comparison

Three local LLM candidates were compared to balance command understanding accuracy and resource consumption. The comparison uses the oracle-text condition on the $N = 43$ command test set to measure intent-to-action correctness without confounding ASR errors. Qwen3 8B achieved the highest $\mathrm{Acc}_{\mathrm{I2A}}$, while Mistral 7B used the least memory but with lower $\mathrm{Acc}_{\mathrm{I2A}}$.

Table 4. Comparison of local LLM models for command understanding

Model	$\mathrm{Acc}_{\mathrm{I2A}}$ (%)	Memory (GB)	Notes
Qwen3 8B	100	6.8	Highest accuracy
Llama3.1 8B	90	6.2	Balanced option
Mistral 7B	80	5.6	Lowest memory

Note: $\mathrm{Acc}_{\mathrm{I2A}}$ is computed on the predefined command test set ($N = 43$) using the intent-to-action correctness definition in the Method section (oracle-text condition).

Smart Building IoT Integration Results

Integration testing was conducted through Home Assistant across multiple device categories (actuators and sensors). The assistant successfully executed voice-driven actions and queries for 33 entities with an overall success rate of 96.67%.

In Table 5, Resp. (s) denotes the observed end-to-end response time (T_{e2e}) as defined in the Method section.

Table 5. IoT integration results across device categories

Device Category	Count	Success (%)	Resp. (s)	Command Example
Weather entity	1	100	5	What's the weather like?
Smart door lock	2	80	10	Lock/unlock the main door
Smart wall switch	8	100	5	Turn on/off the Light 1 Lobby in Lobi Lantai 1
Smart breaker	7	100	5	Turn on/off the Room 12 AC
Room temp. sensors	9	100	4	What is the temperature in Ruang TI 12?
Door status sensors	6	100	4	Is the Room 12 Door open or closed?
Total/Average	**33**	**96.67**	**5.5**	**Multi-modal**

Discussion

Overall, the results indicate that an end-to-end offline voice assistant for smart building management is feasible with acceptable performance in controlled environments. Speech recognition remained reliable in quiet and moderately noisy settings, while high-noise conditions reduced accuracy and may require additional front-end enhancements (e.g., better microphone placement, noise suppression, or domain-adapted prompting). For command understanding, Qwen3 8B provided the best intent-to-action accuracy ($\mathrm{Acc}_{\mathrm{I2A}}$) among the tested local models, suggesting that modern 8B-class models can be practical for local smart building command interpretation when supported by local GPU resources. This finding aligns with recent comparative studies of open-source language models (Aydin et al., 2025). Further optimization through elastic quantization frameworks could reduce memory footprint while maintaining accuracy (Chai et al., 2025). Finally, the high integration success rate across heterogeneous entities demonstrates that a Home Assistant-based approach can offer a flexible and scalable integration layer for offline AIoT assistants.

CONCLUSION

This study demonstrates the feasibility of a privacy-focused, fully offline AIoT voice assistant for smart building management using local LLMs. The proposed architecture combines an edge audio node with an on-premises GPU server hosting containerized AI services (speech-to-text, LLM inference, and text-to-speech) and integrates building entities through Home Assistant, ensuring that voice data and command interpretation remain within the local organizational boundary.

Experimental results in the laboratory testbed show that the system can achieve practical performance for real-world interaction. Speech recognition produced a Word Error Rate (WER) between 5–20% depending on background noise level, and smart building integration reached an overall success rate of 96.67% across 33 IoT entities. Among the evaluated models, Qwen3 8B delivered the highest intent-to-action accuracy ($\mathrm{Acc}_{\mathrm{I2A}}$) on the oracle-text command test set ($N = 43$), supporting the use of modern 8B-class models for local intent interpretation when adequate local compute (e.g., GPU) is available.

Despite these outcomes, performance remains sensitive to acoustic conditions and model/resource constraints. While this study demonstrates feasibility of fully offline systems, federated learning approaches (Pelikan et al., 2025) present complementary directions for scenarios requiring collaborative model improvement. Future work should focus on (i) developing Indonesian-capable TTS and improving Indonesian-capable LLM quality and efficiency for fully on-premises deployments, (ii) expanding interoperability to more heterogeneous device ecosystems (e.g., Zigbee/LoRa/industrial protocols), (iii) scaling evaluation to larger buildings with more users, devices, and complex automation scenarios, (iv) investigating fully edge-deployable and portable voice-assistant satellites using lightweight and quantized STT/LLM/TTS models to reduce reliance on a GPU server, and (v) given emerging acoustic adversarial techniques, future implementations should strengthen security through voice authentication or multi-factor authentication mechanisms and encrypted communication.

ACKNOWLEDGEMENTS

The authors acknowledge the support of Politeknik Negeri Pontianak in enabling this study. This research was supported by the Applied Research (Penelitian Terapan) grant scheme for fiscal year 2025 under Contract No. 188/PL16.B1/PG/2025. The authors further thank the Study Program Coordinator and the Head of the Informatics Engineering Laboratory at Politeknik Negeri Pontianak for facilitating access to the smart-building equipment and laboratory infrastructure used in this work.

REFERENCES

Andreyev, A. (2025). *Quantization for OpenAI's Whisper Models: A Comparative Analysis* (arXiv:2503.09905). arXiv. https://doi.org/10.48550/arXiv.2503.09905

Arora, S., Kachari, K. K., Gupta, S., Saarthi, P., Yadav, A. K., & Patnaik, S. S. (2025). Bringing Llama-3 to the Edge: End-to-End Quantized Conversational AI on Raspberry Pi 5. *2025 IEEE International Conference on Computer Vision and Machine Intelligence (CVMI)*, 1–5. https://doi.org/10.1109/CVMI66673.2025.11337918

Aydin, O., Karaarslan, E., Erenay, F. S., & Bacanin, N. (2025). *Generative AI in Academic Writing: A Comparison of DeepSeek, Qwen, ChatGPT, Gemini, Llama, Mistral, and Gemma* (arXiv:2503.04765). arXiv. https://doi.org/10.48550/arXiv.2503.04765

Beňo, L., Kučera, E., Drahoš, P., & Pribiš, R. (2024). Transforming industrial automation: Voice recognition control via containerized PLC device. *Scientific Reports*, *14*(1), 29387. https://doi.org/10.1038/s41598-024-81172-w

Birkmose, R., Reece, N. M., Norvin, E. H., Bjerva, J., & Zhang, M. (2025). *On-Device LLMs for Home Assistant: Dual Role in Intent Detection and Response Generation* (arXiv:2502.12923). arXiv. https://doi.org/10.48550/arXiv.2502.12923

Bolton, T., Dargahi, T., Belguith, S., Al-Rakhami, M. S., & Sodhro, A. H. (2021). On the Security and Privacy Challenges of Virtual Assistants. *Sensors*, *21*(7), 2312. https://doi.org/10.3390/s21072312

Chai, Y., Kwen, M., Brooks, D., & Wei, G.-Y. (2025). *FlexQuant: Elastic Quantization Framework for Locally Hosted LLM on Edge Devices* (arXiv:2501.07139). arXiv. https://doi.org/10.48550/arXiv.2501.07139

Dwarampudi, A., & Yogi, M. K. (2024). Novel Perspectives in Artificial IoT: Current Trends, Research Directions. *Journal of Cyber Security, Privacy Issues and Challenges*, *3*(1), 22–31. https://doi.org/10.46610/JCSPIC.2024.v03i01.004

Gondi, S., & Pratap, V. (2021). Performance Evaluation of Offline Speech Recognition on Edge Devices. *Electronics*, *10*(21), 2697. https://doi.org/10.3390/electronics10212697

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., Casas, D. de las, Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L. R., Lachaux, M.-A., Stock, P., Scao, T. L., Lavril, T., Wang, T., Lacroix, T., & Sayed, W. E. (2023). *Mistral 7B* (arXiv:2310.06825). arXiv. https://doi.org/10.48550/arXiv.2310.06825

Li, J., chen, C., Pan, L., Azghadi, M. R., Ghodosi, H., & Zhang, J. (2023). *Security and Privacy Problems in Voice Assistant Applications: A Survey* (arXiv:2304.09486). arXiv. https://doi.org/10.48550/arXiv.2304.09486

Li, Q., Ren, X., Wang, Z., Yao, H., He, Y., & Liu, Y. (2025). MetaPipe: Incremental Deployment of Containerized AI Microservices for Edge Clouds. *2025 IEEE/ACM 33rd International Symposium on Quality of Service (IWQoS)*, 1–6. https://doi.org/10.1109/IWQoS65803.2025.11143291

Li, Y., Kim, S., & Sy, E. (2021). *A Survey on Amazon Alexa Attack Surfaces* (arXiv:2102.11442). arXiv. https://doi.org/10.48550/arXiv.2102.11442

Li, Z., Feng, W., Guizani, M., & Yu, H. (2024). *TPI-LLM: Serving 70B-scale LLMs Efficiently on Low-resource Edge Devices* (arXiv:2410.00531). arXiv. https://doi.org/10.48550/arXiv.2410.00531

Maier, E., Doerk, M., Reimer, U., & Baldauf, M. (2023). Digital natives aren't concerned much about privacy, or are they? *I-Com*, *22*(1), 83–98. https://doi.org/10.1515/icom-2022-0041

Ollama. (2026). *Ollama API Documentation (Chat Completions, Structured Outputs, and Tool Calling)*. https://github.com/ollama/ollama/blob/main/docs/api.md

Pelikan, M., Azam, S. S., Feldman, V., Silovsky, J. "Honza," Talwar, K., Brinton, C. G., & Likhomanenko, T. (2025). *Enabling Differentially Private Federated Learning for Speech Recognition: Benchmarks, Adaptive Optimizers and Gradient Clipping* (arXiv:2310.00098). arXiv. https://doi.org/10.48550/arXiv.2310.00098

Shen, Y., Shao, J., Zhang, X., Lin, Z., Pan, H., Li, D., Zhang, J., & Letaief, K. B. (2023). *Large Language Models Empowered Autonomous Edge AI for Connected Intelligence* (arXiv:2307.02779). arXiv. https://doi.org/10.48550/arXiv.2307.02779

Vecino, B. T., Gabrys, A., Matwicki, D., Pomirski, A., Iddon, T., Cotescu, M., & Lorenzo-Trueba, J. (2023). Lightweight End-to-end Text-to-speech Synthesis for low resource on-device applications. *12th ISCA Speech Synthesis Workshop (SSW2023)*, 225–229. https://doi.org/10.21437/SSW.2023-35

Xu, S., & Li, W. (2024). A tool or a social being? A dynamic longitudinal investigation of functional use and relational use of AI voice assistants. *New Media & Society*, *26*(7), 3912–3930. https://doi.org/10.1177/14614448221108112

Yan, C., Ji, X., Wang, K., Jiang, Q., Jin, Z., & Xu, W. (2023). A Survey on Voice Assistant Security: Attacks and Countermeasures. *ACM Computing Surveys*, *55*(4), 1–36. https://doi.org/10.1145/3527153

Ye, S., Du, J., Zeng, L., Ou, W., Chu, X., Lu, Y., & Chen, X. (2024). Galaxy: A Resource-Efficient Collaborative Edge AI System for In-situ Transformer Inference. *IEEE INFOCOM 2024 - IEEE Conference on Computer Communications*, 1001–1010. https://doi.org/10.1109/INFOCOM52122.2024.10621342

Yu, Z., Wang, Z., Li, Y., Gao, R., Zhou, X., Bommu, S. R., Zhao, Y. (Katie), & Lin, Y. (Celine). (2024). EDGE-LLM: Enabling Efficient Large Language Model Adaptation on Edge Devices via Unified Compression and Adaptive Layer Voting. *Proceedings of the 61st ACM/IEEE Design Automation Conference*, 1–6. https://doi.org/10.1145/3649329.3658473

Zhang, L., Wu, S., & Wang, Z. (2025). LoRA-INT8 Whisper: A Low-Cost Cantonese Speech Recognition Framework for Edge Devices. *Sensors*, *25*(17), 5404. https://doi.org/10.3390/s25175404

Zheng, Y., Chen, Y., Qian, B., Shi, X., Shu, Y., & Chen, J. (2025). A Review on Edge Large Language Models: Design, Execution, and Applications. *ACM Computing Surveys*, *57*(8), 1–35. https://doi.org/10.1145/3719664

Zhou, Z., Chen, X., Li, E., Zeng, L., Luo, K., & Zhang, J. (2019). Edge intelligence: Paving the last mile of artificial intelligence with edge computing. *Proceedings of the IEEE*, *107*(8), 1738–1762.